Picture Opcodes

This appendix describes picture opcodes, which are numbers used by the `DrawPicture` procedure to determine what object to draw or what mode to change for subsequent drawing. Your application generally should not read or write picture opcodes directly but should instead use QuickDraw routines (described in the chapter "Pictures" in this book) for generating and processing the opcodes. Picture opcodes are listed here for your application's debugging purposes.

The `Picture` record (described in the chapter "Pictures") begins with a `picSize` field and a `picFrame` field, followed by a variable amount of picture definition data in the form of opcodes. The first opcode in any picture must be the version opcode, followed by the version number of the picture.

Version and Header Opcodes

In a picture created in extended version 2 or version 2 format, the first opcode is the 2-byte `VersionOp` opcode: $0011. This is followed by the 2-byte `Version` opcode: $02FF. With system software version 4.1 or later, the `Version` opcode identifies the picture as an extended version 2 or a version 2 picture, and all subsequent opcodes are read as words (which are word-aligned within the picture). In versions of system software that precede version 4.1, the $02 is read as the version number, then the $FF is read and interpreted as the end-of-picture opcode—for this reason, `DrawPicture` on a pre-4.1 system terminates without drawing any part of an extended version 2 or version 2 picture.

The 2-byte `HeaderOp` opcode ($0C00) follows the `Version` opcode in an extended version 2 or version 2 format picture. The next 24 bytes contain header information. The value of the 2-byte `version` opcode that follows the `HeaderOp` opcode indicates whether the picture is an extended version 2 picture or a version 2 picture: the `Version` opcode has a value of –2 for an extended version 2 picture and a value of –1 for a version 2 picture. The rest of the header for an extended version 2 picture contains resolution information; the rest of the header for a version 2 picture specifies a fixed-point bounding box.

Opcodes that perform drawing commands follow the header information.
The `OpEndPic` opcode ($00FF) signals the end of the picture for an extended version 2 picture or a version 2 picture.

For an example of the version and header opcodes in a decompiled extended version 2 picture, see Listing A-5 on page A-23. For an example of the version and header opcodes in a decompiled version 2 picture, see Listing A-6 on page A-24.

In a version 1 picture, the `VersionOp` opcode has a value of $11, which is followed by a value of $01. For a version 1 picture, QuickDraw parses the remaining drawing opcodes 1 byte at a time; there is no header information in a version 1 picture. An end-of-picture byte ($FF) after the last opcode or data byte in the file signals the end of the picture.

For an example of the version opcodes in a disassembled version 1 picture, see Listing A-7 on page A-25.

Picture Opcode Data Types

The picture opcodes use the data types that are summarized in Table A-1.

Table A-1 Data types for picture opcodes

Data type	Size
`-128..127`	1 byte (signed)
`0..255`	1 byte
`Fixed`	4 bytes
`Integer`	2 bytes
`Long`	4 bytes
`Mode`	2 bytes
`Opcode`	2 bytes
`Pattern`	8 bytes
`Point`	4 bytes
`Poly`	10+ bytes
`Rect`	8 bytes (top, left, bottom, right: integer)
`Rgn`	10+ bytes
`RowBytes`	2 bytes (always an even quantity)

In addition, some picture opcode types, such as `BkPixPat`, may use the `PixMap`, `ColorTable`, and `PixData` data types, which makes the length of these opcodes quite variable. The `PixMap` record and `ColorTable` record are described in the chapter "Color QuickDraw" in this book. The following pseudocode describes the `PixData` data type:

```
PixData: {pseudocode describing the PixData data type}
IF rowBytes < 8 THEN
   data is unpacked;
   data size = rowBytes*(bounds.bottom-bounds.top);
IF rowBytes >= 8 THEN
   data is packed;
   image contains (bounds.bottom-bounds.top) packed scanlines;
   packed scanlines are produced by the PackBits routine;
   each scanline consists of [byteCount] [data];
   IF rowBytes > 250 THEN
      byteCount is a word;
   ELSE
      byteCount is a byte.
END;
```

Opcodes in Pictures

Pictures created with the `OpenPicture` function in a color graphics port use the picture opcodes of the version 2 format. Pictures created with the `OpenCPicture` function use the opcodes of the extended version 2 format. The inclusion of resolution information in the header differentiates the extended version 2 format from the version 2 picture format. The extended version 2 and version 2 formats share the same opcodes, which are listed in Table A-2. The length of the data that follows each 2-byte opcode is listed in this table.

Pictures created with the `OpenPicture` function in a basic graphics port use the opcodes of the version 1 format, which are listed in Table A-3 on page A-18.

The unused opcodes found throughout Table A-2 and Table A-3 are reserved for Apple use. If these opcodes are encountered in pictures, they and their reserved data bytes can simply be skipped. By default, QuickDraw reads and then ignores these opcodes. Because opcodes must be word-aligned in version 2 and extended version 2 pictures, a byte of 0 (zero) data is added after odd-size data.

Note
For opcodes $0100–$7FFF, the amount of data for opcode *$nnXX* = 2 times *nn* bytes. ◆

Table A-2 Opcodes for extended version 2 and version 2 pictures

Opcode	Name	Description	Size (in bytes) of additional data
$0000	`NOP`	No operation	0
$0001	`Clip`	Clipping region	Region size
$0002	`BkPat`	Background pattern	8
$0003	`TxFont`	Font number for text (`Integer`)	2
$0004	`TxFace`	Text's font style (`0..255`)	1
$0005	`TxMode`	Source mode (`Integer`)	2

continued

Table A-2 Opcodes for extended version 2 and version 2 pictures (continued)

Opcode	Name	Description	Size (in bytes) of additional data
$0006	`SpExtra`	Extra space (`Fixed`)	4
$0007	`PnSize`	Pen size (`Point`)	4
$0008	`PnMode`	Pen mode (`Integer`)	2
$0009	`PnPat`	Pen pattern	8
$000A	`FillPat`	Fill pattern	8
$000B	`OvSize`	Oval size (`Point`)	4
$000C	`Origin`	`dh`, `dv` (Integer)	4
$000D	`TxSize`	Text size (`Integer`)	2
$000E	`FgColor`	Foreground color (`Long`)	4
$000F	`BkColor`	Background color (`Long`)	4
$0010	`TxRatio`	Numerator (`Point`), denominator (`Point`)	8
$0011	`VersionOp`	Version (`0..255`)	1
$0012	`BkPixPat`	Background pixel pattern	Variable; see Listing A-1 on page A-17
$0013	`PnPixPat`	Pen pixel pattern	Variable; see Listing A-1 on page A-17
$0014	`FillPixPat`	Fill pixel pattern	Variable; see Listing A-1 on page A-17
$0015	`PnLocHFrac`	Fractional pen position (`Integer`—low word of `Fixed`); if value is not 0.5, pen position is always set to the picture before each text-drawing operation.	2
$0016	`ChExtra`	Added width for nonspace characters (`Integer`)	2
$0017	Reserved for Apple use		Not determined
$0018	Reserved for Apple use		Not determined
$0019	Reserved for Apple use		Not determined
$001A	`RGBFgCol`	Foreground color (`RGBColor`)	6
$001B	`RGBBkCol`	Background color (`RGBColor`)	6

Table A-2 Opcodes for extended version 2 and version 2 pictures (continued)

Opcode	Name	Description	Size (in bytes) of additional data
$001C	`HiliteMode`	Highlight mode flag: no data; this opcode is sent before a drawing operation that uses the highlight mode	0
$001D	`HiliteColor`	Highlight color (`RGBColor`)	6
$001E	`DefHilite`	Use default highlight color; no data; set highlight to default (from low memory)	0
$001F	`OpColor`	Opcolor (`RGBColor`)	6
$0020	`Line`	`pnLoc` (`Point`), `newPt` (`Point`)	8
$0021	`LineFrom`	`newPt` (`Point`)	4
$0022	`ShortLine`	`pnLoc` (`Point`), `dh` (`–128..127`), `dv` (`–128..127`)	6
$0023	`ShortLineFrom`	`dh` (`–128..127`), `dv` (`–128..127`)	2
$0024	Reserved for Apple use	Data length (`Integer`), data	2 + data length
$0025	Reserved for Apple use	Data length (`Integer`), data	2 + data length
$0026	Reserved for Apple use	Data length (`Integer`), data	2 + data length
$0027	Reserved for Apple use	Data length (`Integer`), data	2 + data length
$0028	`LongText`	`txLoc` (`Point`), count (`0..255`), text	5 + text
$0029	`DHText`	`dh` (`0..255`), count (`0..255`), text	2 + text
$002A	`DVText`	`dv` (`0..255`), count (`0..255`), text	2 + text
$002B	`DHDVText`	`dh` (`0..255`), `dv` (`0..255`), count (`0..255`), text	3 + text
$002C	`fontName`	Data length (`Integer`), old font ID (`Integer`), name length (`0..255`), font name*	5 + name length
$002D	`lineJustify`	Operand data length (`Integer`), intercharacter spacing (`Fixed`), total extra space for justification (`Fixed`)†	10
$002E	`glyphState`	Data length (`word`), followed by these 1-byte Boolean values: outline preferred, preserve glyph, fractional widths, scaling disabled	8

continued

Table A-2 Opcodes for extended version 2 and version 2 pictures (continued)

Opcode	Name	Description	Size (in bytes) of additional data
$002F	Reserved for Apple use	Data length (`Integer`), data	2 + data length
$0030	`frameRect`	Rectangle (`Rect`)	8
$0031	`paintRect`	Rectangle (`Rect`)	8
$0032	`eraseRect`	Rectangle (`Rect`)	8
$0033	`invertRect`	Rectangle (`Rect`)	8
$0034	`fillRect`	Rectangle (`Rect`)	8
$0035	Reserved for Apple use	8 bytes of data	8
$0036	Reserved for Apple use	8 bytes of data	8
$0037	Reserved for Apple use	8 bytes of data	8
$0038	`frameSameRect`	Rectangle (`Rect`)	0
$0039	`paintSameRect`	Rectangle (`Rect`)	0
$003A	`eraseSameRect`	Rectangle (`Rect`)	0
$003B	`invertSameRect`	Rectangle (`Rect`)	0
$003C	`fillSameRect`	Rectangle (`Rect`)	0
$003D	Reserved for Apple use		0
$003E	Reserved for Apple use		0
$003F	Reserved for Apple use		0
$0040	`frameRRect`	Rectangle (`Rect`)‡	8
$0041	`paintRRect`	Rectangle (`Rect`)‡	8
$0042	`eraseRRect`	Rectangle (`Rect`)‡	8
$0043	`invertRRect`	Rectangle (`Rect`)‡	8
$0044	`fillRRect`	Rectangle (`Rect`)‡	8
$0045	Reserved for Apple use	8 bytes of data	8
$0046	Reserved for Apple use	8 bytes of data	8
$0047	Reserved for Apple use	8 bytes of data	8
$0048	`frameSameRRect`	Rectangle (`Rect`)	0
$0049	`paintSameRRect`	Rectangle (`Rect`)	0
$004A	`eraseSameRRect`	Rectangle (`Rect`)	0
$004B	`invertSameRRect`	Rectangle (`Rect`)	0
$004C	`fillSameRRect`	Rectangle (`Rect`)	0

Table A-2 Opcodes for extended version 2 and version 2 pictures (continued)

Opcode	Name	Description	Size (in bytes) of additional data
$004D	Reserved for Apple use		0
$004E	Reserved for Apple use		0
$004F	Reserved for Apple use		0
$0050	`frameOval`	Rectangle (`Rect`)	8
$0051	`paintOval`	Rectangle (`Rect`)	8
$0052	`eraseOval`	Rectangle (`Rect`)	8
$0053	`invertOval`	Rectangle (`Rect`)	8
$0054	`fillOval`	Rectangle (`Rect`)	8
$0055	Reserved for Apple use	8 bytes of data	8
$0056	Reserved for Apple use	8 bytes of data	8
$0057	Reserved for Apple use	8 bytes of data	8
$0058	`frameSameOval`	Rectangle (`Rect`)	0
$0059	`paintSameOval`	Rectangle (`Rect`)	0
$005A	`eraseSameOval`	Rectangle (`Rect`)	0
$005B	`invertSameOval`	Rectangle (`Rect`)	0
$005C	`fillSameOval`	Rectangle (`Rect`)	0
$005D	Reserved for Apple use		0
$005E	Reserved for Apple use		0
$005F	Reserved for Apple use		0
$0060	`frameArc`	Rectangle (`Rect`), `startAngle`, `arcAngle`	12
$0061	`paintArc`	Rectangle (`Rect`), `startAngle`, `arcAngle`	12
$0062	`eraseArc`	Rectangle (`Rect`), `startAngle`, `arcAngle`	12
$0063	`invertArc`	Rectangle (`Rect`), `startAngle`, `arcAngle`	12
$0064	`fillArc`	Rectangle (`Rect`), `startAngle`, `arcAngle`	12
$0065	Reserved for Apple use	12 bytes of data	12
$0066	Reserved for Apple use	12 bytes of data	12

continued

Table A-2 Opcodes for extended version 2 and version 2 pictures (continued)

Opcode	Name	Description	Size (in bytes) of additional data
$0067	Reserved for Apple use	12 bytes of data	12
$0068	`frameSameArc`	Rectangle (`Rect`)	4
$0069	`paintSameArc`	Rectangle (`Rect`)	4
$006A	`eraseSameArc`	Rectangle (`Rect`)	4
$006B	`invertSameArc`	Rectangle (`Rect`)	4
$006C	`fillSameArc`	Rectangle (`Rect`)	4
$006D	Reserved for Apple use	4 bytes of data	4
$006E	Reserved for Apple use	4 bytes of data	4
$006F	Reserved for Apple use	4 bytes of data	4
$0070	`framePoly`	Polygon (`Poly`)	Polygon size
$0071	`paintPoly`	Polygon (`Poly`)	Polygon size
$0072	`erasePoly`	Polygon (`Poly`)	Polygon size
$0073	`invertPoly`	Polygon (`Poly`)	Polygon size
$0074	`fillPoly`	Polygon (`Poly`)	Polygon size
$0075	Reserved for Apple use	Polygon (`Poly`)	Polygon size
$0076	Reserved for Apple use	Polygon (`Poly`)	Polygon size
$0077	Reserved for Apple use	Polygon (`Poly`)	Polygon size
$0078	`frameSamePoly`	(Not yet implemented)	0
$0079	`paintSamePoly`	(Not yet implemented)	0
$007A	`eraseSamePoly`	(Not yet implemented)	0
$007B	`invertSamePoly`	(Not yet implemented)	0
$007C	`fillSamePoly`	(Not yet implemented)	0
$007D	Reserved for Apple use		0
$007E	Reserved for Apple use		0
$007F	Reserved for Apple use		0
$0080	`frameRgn`	Region (`Rgn`)	Region size
$0081	`paintRgn`	Region (`Rgn`)	Region size
$0082	`eraseRgn`	Region (`Rgn`)	Region size
$0083	`invertRgn`	Region (`Rgn`)	Region size

Table A-2 Opcodes for extended version 2 and version 2 pictures (continued)

Opcode	Name	Description	Size (in bytes) of additional data
$0084	`fillRgn`	Region (`Rgn`)	Region size
$0085	Reserved for Apple use	Region (`Rgn`)	Region size
$0086	Reserved for Apple use	Region (`Rgn`)	Region size
$0087	Reserved for Apple use	Region (`Rgn`)	Region size
$0088	`frameSameRgn`	(Not yet implemented)	0
$0089	`paintSameRgn`	(Not yet implemented)	0
$008A	`eraseSameRgn`	(Not yet implemented)	0
$008B	`invertSameRgn`	(Not yet implemented)	0
$008C	`fillSameRgn`	(Not yet implemented)	0
$008D	Reserved for Apple use		0
$008E	Reserved for Apple use		0
$008F	Reserved for Apple use		0
$0090	`BitsRect`	`CopyBits` with clipped rectangle	Variable§¶; see Listing A-2 on page A-17
$0091	`BitsRgn`	`CopyBits` with clipped region	Variable§¶; see Listing A-3 on page A-18
$0092	Reserved for Apple use	Data length (`Integer`), data	2 + data length
$0093	Reserved for Apple use	Data length (`Integer`), data	2 + data length
$0094	Reserved for Apple use	Data length (`Integer`), data	2 + data length
$0095	Reserved for Apple use	Data length (`Integer`), data	2 + data length
$0096	Reserved for Apple use	Data length (`Integer`), data	2 + data length
$0097	Reserved for Apple use	Data length (`Integer`), data	2 + data length
$0098	`PackBitsRect`	Packed `CopyBits` with clipped rectangle	Variable§; see Listing A-2 on page A-17
$0099	`PackBitsRgn`	Packed `CopyBits` with clipped rectangle	Variable§; see Listing A-3 on page A-18
$009A	`DirectBitsRect`	`PixMap`, `srcRect`, `dstRect`, `mode` (`Integer`), `PixData`	Variable

continued

Table A-2 Opcodes for extended version 2 and version 2 pictures (continued)

Opcode	Name	Description	Size (in bytes) of additional data
$009B	`DirectBitsRgn`	`PixMap`, `srcRect`, `dstRect`, `mode` (`Integer`), `maskRgn`, `PixData`	Variable
$009C	Reserved for Apple use	Data length (`Integer`), data	2 + data length
$009D	Reserved for Apple use	Data length (`Integer`), data	2 + data length
$009E	Reserved for Apple use	Data length (`Integer`), data	2 + data length
$009F	Reserved for Apple use	Data length (`Integer`), data	2 + data length
$00A0	`ShortComment`	Kind (`Integer`)	2
$00A1	`LongComment`	Kind (`Integer`), size (`Integer`), data	4 + data
$00A2	Reserved for Apple use	Data length (`Integer`), data	2 + data length
.	.	.	.
.	.	.	.
.	.	.	.
$00AF	Reserved for Apple use	Data length (`Integer`), data	2 + data length
$00B0	Reserved for Apple use		0
.	.	.	.
.	.	.	.
.	.	.	.
$00CF	Reserved for Apple use		0
$00D0	Reserved for Apple use	Data length (`Long`), data	4 + data length
.	.	.	.
.	.	.	.
.	.	.	.
$00FE	Reserved for Apple use	Data length (`Long`), data	4 + data length
$00FF	`OpEndPic`	End of picture	2
$0100	Reserved for Apple use	2 bytes of data	2
.	.	.	.
.	.	.	.
.	.	.	.

Table A-2 Opcodes for extended version 2 and version 2 pictures (continued)

Opcode	Name	Description	Size (in bytes) of additional data
$01FF	Reserved for Apple use	2 bytes of data	2
$0200	Reserved for Apple use	4 bytes of data	4
$02FF	`Version`	Version number of picture	2
.	.	.	.
.	.	.	.
.	.	.	.
$0BFF	Reserved for Apple use	22 bytes of data	22
$0C00	`HeaderOp`	For extended version 2: version (`Integer`), reserved (`Integer`), `hRes`, `vRes` (`Fixed`), `srcRect`, reserved (`Long`); for version 2: opcode	24
$0C01	Reserved for Apple use	24 bytes of data	24
.	.	.	.
.	.	.	.
.	.	.	.
$7F00	Reserved for Apple use	254 bytes of data	254
.	.	.	.
.	.	.	.
.	.	.	.
$7FFF	Reserved for Apple use	254 bytes of data	254
$8000	Reserved for Apple use		0
.	.	.	.
.	.	.	.
.	.	.	.
$80FF	Reserved for Apple use		0
$8100	Reserved for Apple use	Data length (`Long`), data	4 + data length
.	.	.	.
.	.	.	.
.	.	.	.

continued

Table A-2 Opcodes for extended version 2 and version 2 pictures (continued)

Opcode	Name	Description	Size (in bytes) of additional data
$8200	`CompressedQuickTime`	Data length (`Long`), data (private to QuickTime)	4 + data length
$8201	`UncompressedQuickTime`	Data length (`Long`), data (private to QuickTime)	4 + data length
$FFFF	Reserved for Apple use	Data length (`Long`), data	4 + data length

* The font name information begins with a word containing the field's data length, followed by a word containing the old font ID, a byte containing the length of the font name, and then the font name itself.

You can extract font names, IDs, and other information from a picture by using the routines described in the chapter "Pictures" in this book.

† For opcode $002D (`lineJustify`), the line justification information contains the line-layout state of the Script Manager so that it can be restored when the picture is played back. It begins with a word containing the field's data length, which should always be 8 bytes. The operands are two fixed-point values, describing the Script Manager's extra character width value and the total extra width that was added to the style run (each `StdText` call) to perform justification.

For example, if the intercharacter spacing were 1 pixel and the total extra width added were 10 pixels, the following hexadecimal bytes would be generated for the picture:

```
2D 00 08 00 01 00 00 00 0A 00 00
```

In this example, the `$002D` opcode is followed by the length word, 00 08, and then the integer part of the intercharacter spacing, 00 01, its fractional part, 00 00, and then the integer part of the total extra spacing, 00 0A, and its fractional part, 00 00.

‡ For opcodes $0040–$0044: rounded rectangles use the setting of the `OvSize` point (refer to opcode $000B).

§ Four opcodes ($0090, $0091, $0098, $0099) are modifications of version 1 opcodes. The first word following the opcode is `rowBytes`. If the high bit of `rowBytes` is set, then it is a pixel map containing multiple bits per pixel; if it is not set, it is a bitmap containing 1 bit per pixel. In general, the difference between version 2 and version 1 formats is that the pixel map replaces the bitmap, a color table has been added, and `pixData` replaces `bitData`.

¶ For opcodes $0090 (`BitsRect`) and $0091 (`BitsRgn`), the data is unpacked. These opcodes can be used only when `rowBytes` is less than 8.

Opcodes $009A (`DirectBitsRect`) and $009B (`DirectBitsRgn`) define direct-pixel pictures, with pixel maps containing three components that directly specify RGB colors. These opcodes allow your application to cut, paste, and store images with up to 32 bits of color information per pixel.

The `DirectBitsRect` and `DirectBitsRgn` opcodes store the `baseAddr` field of the `PixMap` record in a version 2 picture. For compatibility with existing systems, the `baseAddr` field is set to $000000FF. Black-and-white video devices can display pixel maps that are in pictures. On systems without direct-pixel support, opcodes $009A and $009B read a word from the picture and then skip a word of data. The next opcode retrieved from the picture is $00FF, which terminates picture playback. (Note that if you play back a picture on a machine without direct-pixel support, it terminates picture parsing.)

The `DirectBitsRect` opcode is followed by this structure:

```
pixMap:     PixMap;
srcRect:    Rect;    {source rectangle}
dstRect:    Rect;    {destination rectangle}
mode:       Mode;    {transfer mode}
pixData:
```

The `DirectBitsRgn` opcode is followed by this structure:

```
pixMap:     PixMap;
srcRect:    Rect;    {source rectangle}
dstRect:    Rect;    {destination rectangle}
mode:       Mode;    {transfer mode}
maskRgn:    Region;  {region for masking}
pixData:
```

In a picture, the `packType` field of a `PixMap` record specifies the manner in which the pixel data was compressed. To facilitate banding of images when memory is short, all data compression is done on a scan-line basis. The following pseudocode describes the pixel data:

```
PixData:
IF packType = 1 (unpacked) OR rowbytes < 8 THEN
   data is unpacked;
   data size = rowBytes * (bounds.bottom - bounds.top);

IF packType = 2 (drop pad byte) THEN
   the high-order pad byte of a 32-bit direct pixel is
   dropped;
   data size = (3/4) * rowBytes *
   (bounds.bottom - bounds.top);

IF packType > 2 (packed) THEN
   image contains (bounds.bottom - bounds.top) packed
   scan lines;
   each scan line consists of [byteCount] [data];
   IF rowBytes > 250 THEN
      byteCount is a word
   ELSE
      it is a byte
```

Here are the currently defined packing types:

Packing type	Meaning
0	Use default packing
1	Use no packing
2	Remove pad byte—supported only for 32-bit pixels (24-bit data)
3	Run length encoding by `pixelSize` chunks, one scan line at a time—supported only for 16-bit pixels
4	Run length encoding one component at a time, one scan line at a time, red component first—supported only for 32-bit pixels (24-bit data)

For future compatibility, other `packType` values skip scan-line data and draw nothing. Since QuickDraw assumes that pixel map data in memory is unpacked regardless of the `packType` field value, you can use `packType` to tell the picture-recording mechanism what packing technique to use on that data. A `packType` value of 0 in memory indicates that the default packing scheme should be used. (Using the default packing scheme is recommended.) Currently, the default `packType` value for a `pixelSize` value of 16 is type 3; for a `pixelSize` value of 32, it is type 4. Regardless of the setting of `packType` at the time of picture recording, the `packType` value actually used to save the image is recorded in the picture.

Since each scan line of packed data is preceded by a byte count, `packSize` is not used and must be 0 for future compatibility.

When the pixel type is direct, `cmpCount * cmpSize` is less than or equal to `pixelSize`. For storing 24-bit data in a 32-bit pixel, set `cmpSize` to 8 and `cmpCount` to 3. If you set `cmpCount` to 4, then the high byte is compressed by packing scheme 4 and stored in the picture.

The `OpenCPicture` function lets your application create a version 2 format picture and include rectangle and resolution information, which is stored in the version 2 picture header. The `OpenCPicture` function is described in the chapter “Pictures.”

The `HeaderOp` information is passed to the `OpenCPicture` function as an `OpenCPicParams` record, which is described in the chapter “Pictures” in this book.

The pseudocode in Listing A-1 illustrates the data for the `BkPixPat`, `PnPixPat`, and `FillPixPat` opcodes.

Listing A-1 Data for the `BkPixPat`, `PnPixPat`, and `FillPixPat` opcodes

```
IF patType = ditherPat
THEN
   PatType:    word;       {pattern type = 2}
   Pat1Data:   Pattern;    {old pattern data}
   RGB:        RGBColor;   {desired RGB for pattern}
ELSE
   PatType:    word;       {pattern type = 1}
   Pat1Data:   Pattern;    {old pattern data}
   PixMap:     PixMap;
   ColorTable: ColorTable;
   PixData:    PixData;
END;
```

The pseudocode in Listing A-2 illustrates the data is stored in the `BitsRect` and `PackBitsRect` opcodes.

Listing A-2 Data for the `BitsRect` and `PackBitsRect` opcodes

```
   PixMap:     PixMap;     {pixel map}
   ColorTable: ColorTable; {ColorTable record}
   srcRect:    Rect;       {source rectangle}
   dstRect:    Rect;       {destination rectangle}
   mode:       Word;       {transfer mode (may include }
                           { new transfer modes)}
   PixData:    PixData;
```

The pseudocode in Listing A-3 illustrates the data is stored in the `BitsRgn` and `PackBitsRgn` opcodes.

Listing A-3 Data for the `BitsRgn` and `PackBitsRgn` opcodes

```
pixMap:     PixMap;
colorTable: ColorTable;
srcRect:    Rect;          {source rectangle}
dstRect:    Rect;          {destination rectangle}
mode:       Word;          {transfer mode (may }
                           { include new modes)}
maskRgn:    Rgn;           {region for masking}
pixData:    PixData;
```

Pictures created with the `OpenPicture` function in a basic graphics port use the opcodes of the version 1 format, as listed in Table A-3. This size of data that follows each opcode is also listed in this table. Version 1 pictures are limited to 32 KB.

Table A-3 Opcodes for version 1 pictures

Opcode	Name	Description	Size (in bytes) of additional data
$00	`NOP`	No operation	0
$01	`ClipRgn`	Clipping region	Region size
$02	`BkPat`	Background pattern	8
$03	`TxFont`	Font number for text (`Integer`)	2
$04	`TxFace`	Text's font style (`0..255`)	1
$05	`TxMode`	Source mode (`Integer`)	2
$06	`SpExtra`	Extra space (`Fixed`)	4
$07	`PnSize`	Pen size (`Point`)	4
$08	`PnMode`	Pen mode (`Integer`)	2
$09	`PnPat`	Pen pattern	8
$0A	`FillPat`	Fill pattern	8
$0B	`OvSize`	Oval size (`Point`)	4
$0C	`Origin`	`dh` (`Integer`), `dv` (`Integer`)	4
$0D	`TxSize`	Text size (`Integer`)	2
$0E	`FgColor`	Foreground color (`Long`)	4
$0F	`BkColor`	Background color (`Long`)	4

Table A-3 Opcodes for version 1 pictures (continued)

Opcode	Name	Description	Size (in bytes) of additional data
$10	`TxRatio`	Numerator (`Point`), denominator (`Point`)	8
$11	`picVersion`	Version (`0..255`)	1
$20	`Line`	`pnLoc` (`Point`), `newPt` (`Point`)	8
$21	`LineFrom`	`newPt` (`Point`)	4
$22	`ShortLine`	`pnLoc` (`Point`), `dh` (`–128..127`), `dv` (`–128..127`)	6
$23	`ShortLineFrom`	`dh` (`–128..127`), `dv` (`–128..127`)	2
$28	`LongText`	`txLoc` (`Point`), count (`0..255`), text	5 + text
$29	`DHText`	`dh` (`0..255`), count (`0..255`), text	2 + text
$2A	`DVText`	`dv` (`0..255`), count (`0..255`), text	2 + text
$2B	`DHDVText`	`dh` (`0..255`), `dv` (`0..255`), count (`0..255`), text	3 + text
$30	`frameRect`	Rectangle (`Rect`)	8
$31	`paintRect`	Rectangle (`Rect`)	8
$32	`eraseRect`	Rectangle (`Rect`)	8
$33	`invertRect`	Rectangle (`Rect`)	8
$34	`fillRect`	Rectangle (`Rect`)	8
$38	`frameSameRect`	Rectangle (`Rect`)	0
$39	`paintSameRect`	Rectangle (`Rect`)	0
$3A	`eraseSameRect`	Rectangle (`Rect`)	0
$3B	`invertSameRect`	Rectangle (`Rect`)	0
$3C	`fillSameRect`	Rectangle (`Rect`)	0
$40	`frameRRect`	Rectangle (`Rect`)*	8
$41	`paintRRect`	Rectangle (`Rect`)*	8
$42	`eraseRRect`	Rectangle (`Rect`)*	8
$43	`invertRRect`	Rectangle (`Rect`)*	8
$44	`fillRRect`	Rectangle (`Rect`)*	8
$48	`frameSameRRect`	Rectangle (`Rect`)	0
$49	`paintSameRRect`	Rectangle (`Rect`)	0
$4A	`eraseSameRRect`	Rectangle (`Rect`)	0

continued

Table A-3 Opcodes for version 1 pictures (continued)

Opcode	Name	Description	Size (in bytes) of additional data
$4B	`invertSameRRect`	Rectangle (`Rect`)	0
$4C	`fillSameRRect`	Rectangle (`Rect`)	0
$50	`frameOval`	Rectangle (`Rect`)	8
$51	`paintOval`	Rectangle (`Rect`)	8
$52	`eraseOval`	Rectangle (`Rect`)	8
$53	`invertOval`	Rectangle (`Rect`)	8
$54	`fillOval`	Rectangle (`Rect`)	8
$58	`frameSameOval`	Rectangle (`Rect`)	0
$59	`paintSameOval`	Rectangle (`Rect`)	0
$5A	`eraseSameOval`	Rectangle (`Rect`)	0
$5B	`invertSameOval`	Rectangle (`Rect`)	0
$5C	`fillSameOval`	Rectangle (`Rect`)	0
$60	`frameArc`	Rectangle (`Rect`), `startAngle`, `arcAngle`	12
$61	`paintArc`	Rectangle (`Rect`), `startAngle`, `arcAngle`	12
$62	`eraseArc`	Rectangle (`Rect`), `startAngle`, `arcAngle`	12
$63	`invertArc`	Rectangle (`Rect`), `startAngle`, `arcAngle`	12
$64	`fillArc`	Rectangle (`Rect`), `startAngle`, `arcAngle`	12
$68	`frameSameArc`	Rectangle (`Rect`)	4
$69	`paintSameArc`	Rectangle (`Rect`)	4
$6A	`eraseSameArc`	Rectangle (`Rect`)	4
$6B	`invertSameArc`	Rectangle (`Rect`)	4
$6C	`fillSameArc`	Rectangle (`Rect`)	4
$70	`framePoly`	Polygon (`Poly`)	Polygon size
$71	`paintPoly`	Polygon (`Poly`)	Polygon size
$72	`erasePoly`	Polygon (`Poly`)	Polygon size
$73	`invertPoly`	Polygon (`Poly`)	Polygon size
$74	`fillPoly`	Polygon (`Poly`)	Polygon size
$78	`frameSamePoly`	(Not yet implemented)	0

Table A-3 Opcodes for version 1 pictures (continued)

Opcode	Name	Description	Size (in bytes) of additional data
$79	`paintSamePoly`	(Not yet implemented)	0
$7A	`eraseSamePoly`	(Not yet implemented)	0
$7B	`invertSamePoly`	(Not yet implemented)	0
$7C	`fillSamePoly`	(Not yet implemented)	0
$80	`frameRgn`	Region (`Rgn`)	Region size
$81	`paintRgn`	Region (`Rgn`)	Region size
$82	`eraseRgn`	Region (`Rgn`)	Region size
$83	`invertRgn`	Region (`Rgn`)	Region size
$84	`fillRgn`	Region (`Rgn`)	Region size
$88	`frameSameRgn`	(Not yet implemented)	0
$89	`paintSameRgn`	(Not yet implemented)	0
$8A	`eraseSameRgn`	(Not yet implemented)	0
$8B	`invertSameRgn`	(Not yet implemented)	0
$8C	`fillSameRgn`	(Not yet implemented)	0
$90	`BitsRect`	`CopyBits` with clipped rectangle	Variable[†‡]; see Listing A-2 on page A-17
$91	`BitsRgn`	`CopyBits` with clipped region	Variable[†‡]; see Listing A-3 on page A-18
$98	`PackBitsRect`	Packed `CopyBits` with clipped rectangle	Variable[†]; see Listing A-2 on page A-17
$99	`PackBitsRgn`	Packed `CopyBits` with clipped rectangle	Variable[†]; see Listing A-3 on page A-18
$A0	`ShortComment`	Kind (`Integer`)	2
$A1	`LongComment`	Kind (`Integer`), size (`Integer`), data	4 + data
$FF	`EndOfPicture`	End of picture	0

[*] For opcodes $40–$44: rounded rectangles use the setting of the `OvSize` point (refer to opcode $0B).
[†] In general, the difference between version 2 and version 1 formats is that the pixel map replaces the bitmap, a color table has been added, and `pixData` replaces `bitData`.
[‡] For opcodes $90 (`BitsRect`) and $91 (`BitsRgn`), the data is unpacked. These opcodes can only be used when `rowBytes` is less than 8.

A Sample Extended Version 2 Picture

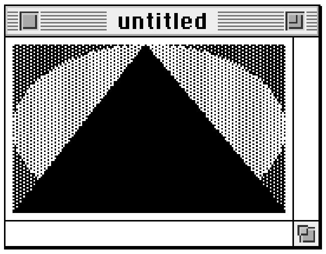
The chapter “Pictures” in this book describes how to use the `OpenCPicture` function to create and display extended version 2 pictures. Listing A-4 illustrates how to use `OpenCPicture`.

Listing A-4 Creating and drawing an extended version 2 picture

```
FUNCTION MyCreateAndDrawPict(pFrame: Rect): PicHandle;
VAR
   myOpenCPicParams: OpenCPicParams;
   myPic:            PicHandle;
   trianglePoly:     PolyHandle;
BEGIN
   WITH myOpenCPicParams DO BEGIN
      srcRect := pFrame;
      hRes := gHRes;       {$00480000 for 72 dpi}
      vRes := gVRes;       {$00480000 for 72 dpi}
      version := - 2;      {always set this field to -2}
      reserved1 := 0;      {this field is unused}
      reserved2 := 0;      {this field is unused}
   END;
   myPic := OpenCPicture(myOpenCPicParams);  {start creating the picture}
   ClipRect(pFrame);                      {always set a valid clip region}
   FillRect(pFrame,dkGray);   {create a dark gray rectangle for background}
   FillOval(pFrame,ltGray);   {overlay the rectangle with a light gray oval}
   trianglePoly := OpenPoly;  {start creating a triangle}
   WITH pFrame DO BEGIN
      MoveTo(left,bottom);
      LineTo((right - left) DIV 2,top);
      LineTo(right,bottom);
      LineTo(left,bottom);
   END;
   ClosePoly;                 {finish the triangle}
   PaintPoly(trianglePoly);   {paint the triangle}
   KillPoly(trianglePoly);    {dispose of the memory for the triangle}
   ClosePicture;              {finish the picture}
   DrawPicture(myPic,pFrame);       {draw the picture}
   MyCreateAndDrawPict := myPic;
END;
```

Figure A-1 shows the picture created by Listing A-4.

Figure A-1 A picture



The QuickDraw drawing commands issued between `OpenCPicture` and the `ClosePicture` procedure in Listing A-4 are saved in memory as a `Picture` record containing a `picSize` field, a `picFrame` field, and an array of picture opcodes; an application can also save this information in a resource of type `'PICT'`. The `DrawPicture` procedure reads these opcodes when drawing the picture.

For debugging purposes, you might find it helpful to examine the opcodes for a picture. Listing A-5 shows the extended version 2 picture in Figure A-1 after it is saved in a `'PICT'` resource and then decompiled with the DeRez decompiler.

Listing A-5 A decompiled extended version 2 picture

```
data 'PICT' (128) {
$"0078"     /* picture size; don't use this value for picture size */
$"0000 0000 006C 00A8"  /* bounding rectangle of picture at 72 dpi */
$"0011"     /* VersionOp opcode; always $0011 for extended version 2 */
$"02FF"     /* Version opcode; always $02FF for extended version 2 */
$"0C00"     /* HeaderOp opcode; always $0C00 for extended version 2 */
            /* next 24 bytes contain header information */
   $"FFFE"  /* version; always -2 for extended version 2 */
   $"0000"  /* reserved */
   $"0048 0000"         /* best horizontal resolution: 72 dpi */
   $"0048 0000"         /* best vertical resolution: 72 dpi */
   $"0002 0002 006E 00AA"  /* optimal source rectangle for 72 dpi horizontal
                              and 72 dpi vertical resolutions */
   $"0000"  /* reserved */
$"001E"     /* DefHilite opcode to use default hilite color */
$"0001"     /* Clip opcode to define clipping region for picture */
   $"000A"  /* region size */
   $"0002 0002 006E 00AA"  /* bounding rectangle for clipping region */
$"000A"     /* FillPat opcode; fill pattern specified in next 8 bytes */
```

```
   $"77DD 77DD 77DD 77DD"  /* fill pattern */
$"0034"     /* fillRect opcode; rectangle specified in next 8 bytes */
   $"0002 0002 006E 00AA"  /* rectangle to fill */
$"000A"     /* FillPat opcode; fill pattern specified in next 8 bytes */
   $"8822 8822 8822 8822"  /* fill pattern */
$"005C"     /* fillSameOval opcode */
$"0008"     /* PnMode opcode */
$  "0008"   /* pen mode data */
$"0071"     /* paintPoly opcode */
   $"001A"  /* size of polygon */
   $"0002 0002 006E 00AA"  /* bounding rectangle for polygon */
   $"006E 0002 0002 0054 006E 00AA 006E 0002"   /* polygon points */
$"00FF"     /* OpEndPic opcode; end of picture */
}
```

A Sample Version 2 Picture

The chapter "Pictures" in this book describes how to use the `OpenPicture` function, which creates version 2 pictures in color graphics ports. Figure A-1 on page A-23 shows a picture created with the `OpenCPicture` function using the code in Listing A-4 on page A-22. If the `OpenPicture` function were used instead of `OpenCPicture`, the same picture would be drawn, but the picture would use picture opcodes for the version 2 format instead of the extended version 2 format. The major difference between formats lies in the header information after the `HeaderOp` opcode.

Listing A-6 shows what happens when the picture in Figure A-1 is created in version 2 format, saved in a `'PICT'` resource, and then decompiled with the DeRez decompiler.

Listing A-6 A decompiled version 2 picture

```
data 'PICT' (129) {
$"0078"     /* picture size; don't use this value for picture size */
$"0002 0002 006E 00AA"  /* bounding rectangle of picture */
$"0011"     /* VersionOp opcode; always $0011 for version 2 */
$"02FF"     /* Version opcode; always $02FF for version 2 */
$"0C00"     /* HeaderOp opcode; always $0C00 for version 2 */
            /* next 24 bytes contain header information */
   $"FFFF FFFF"   /* version; always -1 (long) for version 2 */
   $"0002 0000 0002 0000 00AA 0000 006E 0000"   /* fixed-point bounding
                                                   rectangle for picture */
   $"0000 0000"   /* reserved */
$"001E"     /* DefHilite opcode to use default hilite color */
$"0001"     /* Clip opcode to define clipping region for picture */
   $"000A"  /* region size */
   $"0002 0002 006E 00AA"  /* bounding rectangle for clipping region */
$"000A"     /* FillPat opcode; fill pattern specifed in next 8 bytes */
   $"77DD 77DD 77DD 77DD"  /* fill pattern */
$"0034"     /* fillRect opcode; rectangle specified in next 8 bytes */
   $"0002 0002 006E 00AA"  /* rectangle to fill */
$"000A"     /* FillPat opcode; fill pattern specified in next 8 bytes */
   $"8822 8822 8822 8822"  /* fill pattern */
$"005C"     /* fillSameOval opcode */
$"0008"     /* PnMode opcode */
$  "0008"   /* pen mode data */
$"0071"     /* paintPoly opcode */
   $"001A"  /* size of polygon */
   $"0002 0002 006E 00AA"  /* bounding rectangle for polygon */
   $"006E 0002 0002 0054 006E 00AA 006E 0002"   /* polygon points */
$"00FF"     /* OpEndPic opcode; end of picture */
}
```

A Sample Version 1 Picture

Pictures created by the `OpenPicture` function on computers without Color QuickDraw, or when the current graphics port is a basic graphics port, are created in version 1 format. The code in Listing A-7 shows what happens when the picture in Figure A-1 on page A-23 is created in version 1 format, saved in a `'PICT'` resource, and then decompiled with the DeRez decompiler.

Listing A-7 A decompiled version 1 picture

```
data 'PICT' (130) {
$"004F"     /* picture size; this value is reliable for version 1 pictures */
$"0002 0002 006E 00AA"  /* bounding rectangle of picture */
$"11"       /* picVersion opcode for version 1 */
   $"01"    /* version number 1 */
$"01"       /* ClipRgn opcode to define clipping region for picture */
   $"000A"  /* region size */
   $"0002 0002 006E 00AA"  /* bounding rectangle for region */
$"0A"       /* FillPat opcode; fill pattern specified in next 8 bytes */
   $"77DD 77DD 77DD 77DD"  /* fill pattern */
$"34"       /* fillRect opcode; rectangle specified in next 8 bytes */
   $"0002 0002 006E 00AA"  /* rectangle to fill */
$"0A"       /* FillPat opcode; fill pattern specified in next 8 bytes */
   $"8822 8822 8822 8822"  /* fill pattern */
$"5C"       /* fillSameOval opcode */
$"71"       /* paintPoly opcode */
   $"001A"  /* size of polygon */
   $"0002 0002 006E 00AA"  /* bounding rectangle for polygon */
   $"006E 0002 0002 0054 006E 00AA 006E 0002"   /* polygon points */
$"FF"       /* EndOfPicture opcode; end of picture */
}
```